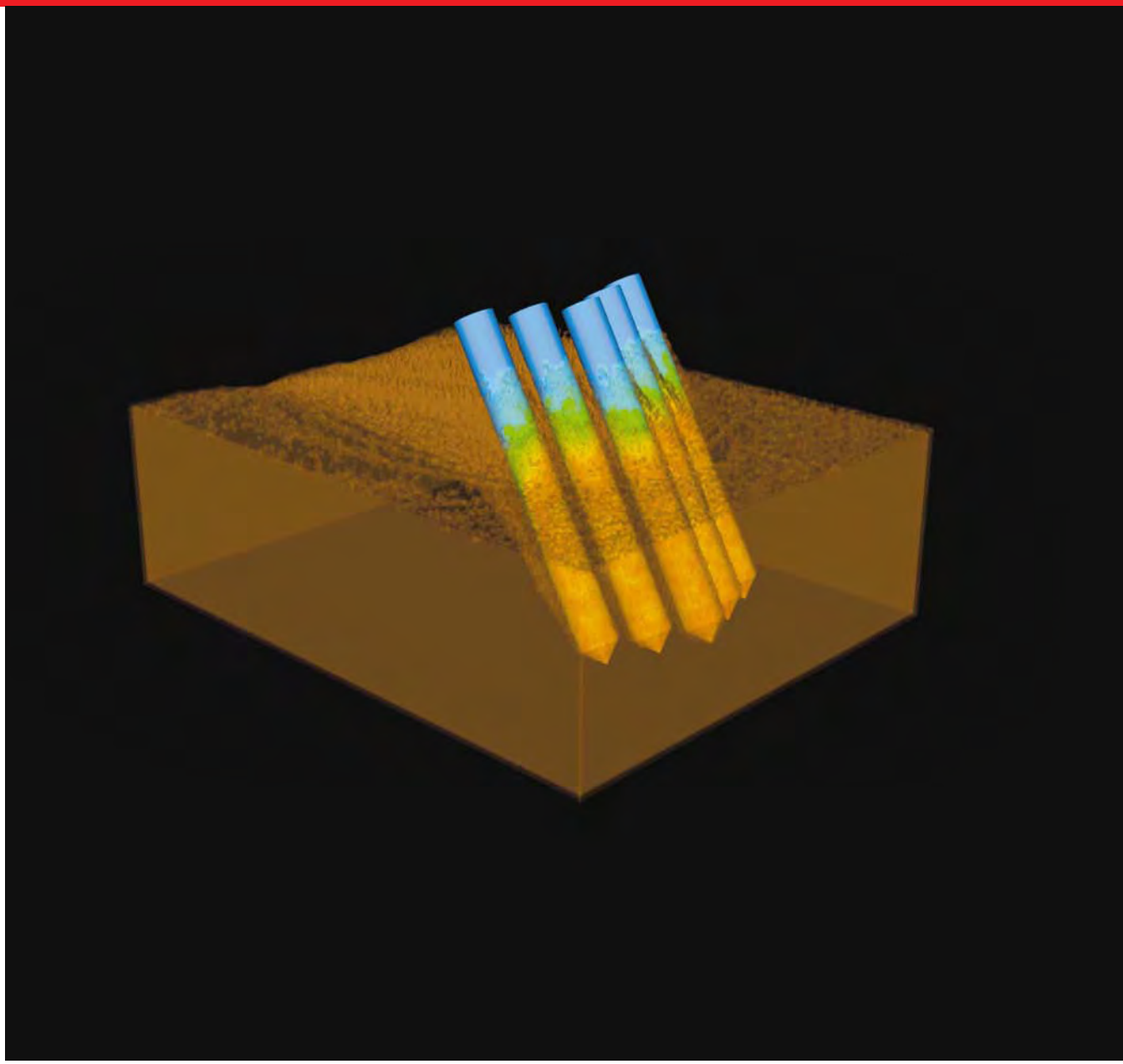
CEWES Major Shared Resource Center

MSRC

JOURNAL

FALL 1998



Inside This Issue:

MetaCenter • Distance Learning • Turbulence Modeling • Virtual Proving Ground • Programming Tools • Information Display • Programming Models • Scientific Visualization • Performance

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1998		**00-00-1998 to 00-00-1998**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
CEWES MSRC. Fall 1998	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **U.S. Army Engineer Research and Development Center,ATTN: ERDC MSRC HPC Resource Center,3909 Halls Ferry Road,Vicksburg,MS,39180-6199**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**24**	

WHO WE ARE

The U.S. Army Corps of Engineers, Waterways Experiment Station (CEWES) Major Shared Resource Center (MSRC) is part of the High Performance Computing Modernization Program (HPCMP) and is located in the Information Technology Laboratory at CEWES in Vicksburg, MS. As a world-class facility, the CEWES MSRC employs a technical staff to provide full-spectrum computational support for DoD researchers, from Help Desk assistance to one-on-one collaboration. More than 4,000 computational scientists and engineers are involved in the HPCMP with immediate access to DoD HPC capabilities, regardless of their locations across the nation, via the Defense Research and Engineering Network.

Other services include a diverse and well-equipped Scientific Visualization Center for visualization expertise and capability. In addition, the Programming Environment and Training component provides for transfer of cutting-edge HPC technology and training and development activities for acquiring HPC skills.

EDITORIAL STAFF

✔ Outreach Coordinator
Mary L. Hampton

✔ Chief Editor
Michelle Morgan Brown

✔ Assistant Editor
Mary Gabb

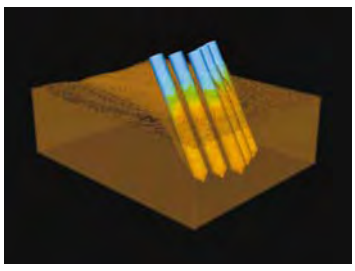
✔ Technical Editor
John E. West

Design and layout provided by the Visual Production Center, Information Technology Laboratory, U.S. Army Engineer Waterways Experiment Station.

CONTENTS

THE FRONT COVER:

This image shows a four-tined mine plow configuration moving from left to right through a soil mass of 10 million particles. Colors on the plow indicate pressures exerted on the tines from low (blue) to high (orange/red). Recent computational developments have increased the number of particles that can be simulated using this technology from a few hundred thousand to 10 million particles. The very large-scale discrete element model provides a virtual laboratory for evaluation of vehicle-terrain interaction effects that previously required expensive laboratory and field studies.



CEWES MSRC and Partners Embark on MetaCenter Project

Judith Utley

The CEWES MSRC, in collaboration with the Aeronautical Systems Command (ASC) MSRC, has created an IBM SP-based MetaCenter. This MetaCenter allows users at each site to submit jobs to their local batch queuing system with the scheduler, depending on job requirements and system load, scheduling the job on the most appropriate IBM SP system. The underlying technology was developed by NASA and used in its implementation of a MetaCenter between the NASA Ames Research Center and NASA Langley Research Center. The NASA MetaCenter existed from October 1996 until the IBM SP systems were decommissioned in February 1998.

The Portable Batch System (PBS) developed at NASA Ames and now supported by MRJ Technology Solutions provides the glue that bonds the two sites and makes the MetaCenter work. PBS operates in networked UNIX environments and provides very flexible batch scheduling for both batch and interactive work. A separate job scheduler runs on each system, making it easy to implement and modify site-specific policies. The job scheduler, external to PBS, determines which jobs to run when and on which system. Each system works independently so long as the jobs queued and jobs running allow it to keep busy (i.e., above a configurable utilization threshold). Once the work load drops below this predefined utilization threshold, the job scheduler begins peer scheduling.

When a system needs to find work to increase utilization, the job scheduler asks other systems in the MetaCenter (its fipeersf for a list of their queued jobs on that remote system. It proceeds through this job list, checking each candidate to see which jobs, if any, it can run (based on local scheduling policies). Once an eligible job is found, the scheduler uses a PBS *movejob* request to pull the job from the peer PBS server to the local server. The PBS scheduler checks several job requirements such as user-requested attributes (e.g., a particular site or software package), the number of nodes or the type of nodes requested, whether the time requested fits policies, and the presence of an active user account on the local system. If a job passes all of these tests, it is moved to the local system and run. The peer scheduler continues looking for jobs until the utilization increases above the predefined threshold.

Once a job has been moved, the peer scheduler initiates any needed file staging operations to move files required by the job from one system to the other. Users include file-staging directives in the job submission script. These directives also allow users to tell PBS where they would like to have their output files placed. Otherwise, these output files are returned to the remote fipeer from which the job originated. More information about PBS can be found on the Internet at *http://science.nas.nasa.gov/Software/PBS/*.

MetaCenter

The CEWES/ASC MetaCenter project will offer tremendous potential for the CEWES and ASC user communities. With the work load balanced across both systems, users can expect to see improved turnaround time for their jobs. The MetaCenter will also offer more flexibility for the user community. Users will be able to submit a job at their fihome site, and the job scheduler will run the job where it is most appropriate. Users will not have to keep up with which site has a particular software package, nor will every site need to purchase and maintain all software packages.

During operating system upgrades, scheduled maintenance, or almost any system downtime, users may continue to submit jobs to a local server and have their jobs run automatically at the remote site, providing much less interruption to their schedules. By coordinating these outages, everyone profits, users as well as system personnel, as one site learns from the upgrade experiences of the other. The MetaCenter staff also benefits as the combined staff of the two MSRCs provides a larger base of expertise. Working together, the two sites can provide a more dynamic and user friendly environment.

Once this first SP MetaCenter project is in production and the benefits are readily seen, other sites may wish to take advantage of the technology or even participate in the effort. Additional systems at each site may be included in the MetaCenter as well. This endeavor should provide the foundation for expansion to other similar initiatives at other agencies and educational institutions.

Virtual Proving Ground

Mine Plowing Simulations on the CEWES MSRC HPC Systems

David A. Horner, Ph.D.
Alex R. Carrillo

A goal of the U.S. Army is to accomplish Research and Development tasks through modeling and simulation applications. As an example, virtual proving grounds (VPGs) are being developed to assist in the evaluation of fielding new vehicles and vehicle-mounted weapon systems. Through modeling and simulation of the vehicle components, VPGs will be used to accomplish the many tasks required to field a new system from concept to prototype. The U.S. Army Corps of Engineers, Waterways Experiment Station (CEWES) has initiated research efforts to support the development of VPGs. One such effort involves one of the top priorities of Engineer Regiments, the fiGrizzly f The Grizzly is a breaching vehicle capable of clearing wire obstacles, anti-mine ditches, log cribs, rubble, and, most importantly, mines (to a depth of 12 in. at up to 3 mph). Currently mine clearing is a major obstacle to rapid advance forces.

For the Grizzly to meet its potential, the plowing system must maintain proper depth control without intervention of the operators. Present designs use a depth control guided

by forces acting on the plow tines (Figures 1 and 2). The deeper the plow, the higher the force on the tines. The force-depth relationship is obtained from a pre-breach calibration. However, local variations in the soil may cause the plow to dive too deep, causing the vehicle to stall, or to dive too shallow, causing mines to be missed or detonated. Designing the feedback mechanism that adjusts the plow depth for tine force changes to be less sensitive to small soil strength changes requires field experiments. Field experiments are also required to develop alternative systems for plow depth control and alternative plow design. Such experiments are expensive, especially at the multiple field sites needed to verify the general effectiveness of the plow. A virtual proving ground will greatly benefit the design process by allowing the designer to obtain realistic, simulated plow-tine performance data that supplements limited field experiments. Computer-generated experimental environments could be adjusted to replicate a variety of ground conditions representing diverse geographic locations. Multiple simulations can be run through identical soil to isolate effects of operational parameters. Sensitivity to variations in soil conditions could be determined by systematically varying the statistical distribution of soil strength.

The "Grizzly" is a breaching vehicle capable of clearing wire obstacles, anti-mine ditches, log cribs, rubble, and, most importantly, mines

One of the keys to a successful vehicle VPG is fast and realistic simulation of interaction at the vehicle-soil interface. Typically, the vehicle-soil interface can involve large discontinuous deformations of the soil mass. Soil plowing as used in the clearing of land mines, tire/track sinkage, and the development of traction in loose soils during off-road movement are examples of items that can cause the large soil deformations. CEWES has been engaged in improving large deformation modeling as it relates to mobility problems for the past 5 years. Two major deficiencies in existing simulation technology are evident. First, large deformation in soils is poorly understood and models designed for other engineering materials are not applicable to soils. Second, numerical methods based on finite difference or finite element techniques do not capture discontinuous soil deformation. Therefore, discrete particle methods were adopted. Unfortunately, particle

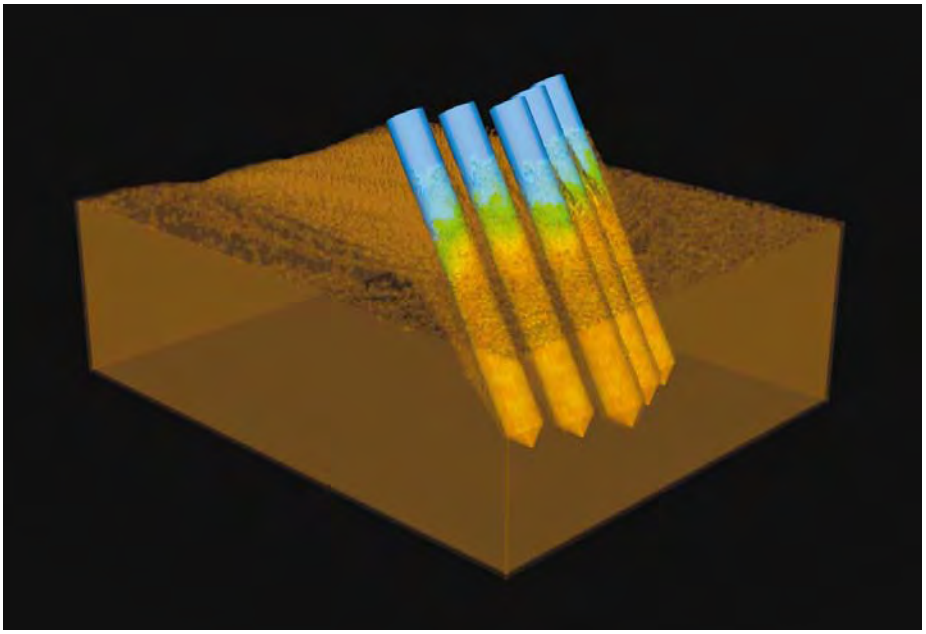


Figure 1. Soil-Tine Interaction. Four-tined mine plow configuration moving left to right through a soil mass of 10 million particles which has been rendered partially transparent. Colors on the plow indicate pressures exerted on the tines from low (blue) to high (orange/red).

Virtual Proving Ground

methods require significant computational resources; thus, the entire modeling approach is built around the existence of high performance computing resources.

Currently, large distributed memory parallel computing resources are being utilized at the CEWES MSRC to study the soil-tine interaction of the Grizzly. Figure 1 shows the soil-tine interaction for a plowing simulation. The simulation involves 10 million particles - the largest particle simulation being run with non-trivial boundary conditions - and was run on 192 processors of the Cray T3E. Similar runs have been performed on the IBM SP. Capability tests have been run with as many as 40 million particles. Available memory will allow runs with up to 100 million particles, but the required run time makes this scale of simulation prohibitive.

The Grizzly has the potential to greatly enhance the effectiveness of rapid response forces by simplifying breaching operations and reducing casualties. Ensuring the Grizzly™s effectiveness under varied field situations requires extensive field experiments. The development of a virtual proving ground will enhance the interpretation of those field results, greatly supplementing the experimental database. A capability will be established to rapidly simulate plow performance in new operational environments prior to field deployment. Current work in the development of a smoothed particle method has the potential to greatly expand vehicle-terrain simulations in Department of Defense (DoD) applications by broadening problem sizes. The Grizzly mine plow represents an opportunity to bring these methods to a developmental state needed for widespread use of particle methods for soil-vehicle interaction.

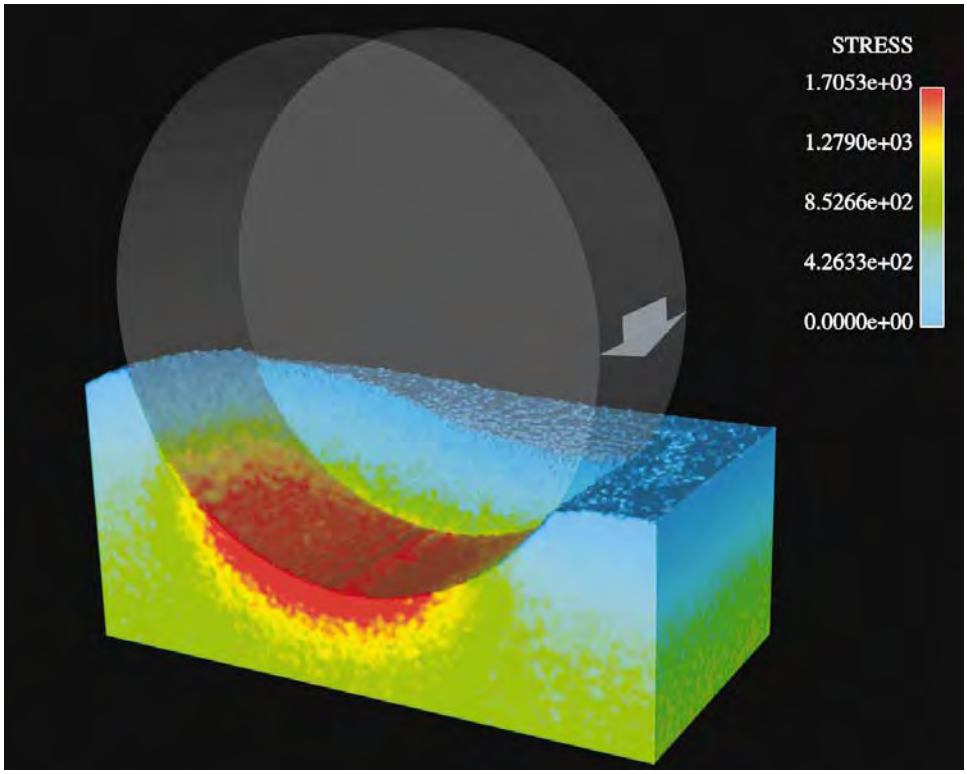


FIGURE 2. WHEEL OF MINE PLOW. Pictured is an idealized wheel, rendered partially transparent, turning into a 10 million particle soil mass in the direction of the arrow. Colors indicate stress exerted on individual particles in the soil from blue (low) to red (high).

Use of the TANGO Interactive Collaboratory Tool in the CEWES MSRC PET Program

David E. Bernholdt, Ph.D.
Nancy J. McCracken, Ph.D.
Marek Podgorny, Ph.D.
Geoffrey C. Fox, Ph.D.

The DoD Research and Development community is widely distributed geographically. This presents quite a challenge to providing researchers with access to training on high performance computing systems and technologies and to professional education in computational science, two important aspects of the Programming Environment and Training (PET) program. Distance education is not a new idea, but it often requires specialized equipment (such as satellite uplinks and specialized videoconferencing systems) or does not provide the level of interactivity many students desire (i.e., videotaped lectures). However, it is now becoming possible to routinely deliver courses in real-time over the Internet. TANGO Interactive, developed at the Northeast Parallel Architectures Center at Syracuse University, an academic partner in the CEWES MSRC PET Program, is a network-based collaborative tool which is currently being used for remote education and training activities and will soon be made available to users of the CEWES MSRC for more general collaborative use.

As far as TANGO is concerned, distance learning is just a special case of electronic collaboration . . .

TANGO provides a framework that allows applications to be shared remotely over the network, not just for education, but for any kind of remote collaboration. The TANGO system has a client-server architecture in which the clients consist of a control application and a variety of shared applications. The control application handles administrative functions, such as launching applications and tracking which users are sharing each application. Collaborative applications send fievents to the TANGO Server, which then rebroadcasts them to other clients sharing the application. Apart from a few basic functions, the developer of the application is free to define whatever shared events that are appropriate to a particular application. If communication performance is critical, clients may also communicate directly, bypassing the server. TANGO also provides a basic form of fifloo controlf by keeping track of a simple master/slave flag for each instance of a shared application and providing a mechanism to grant and relinquish master status. Again, the application developer can interpret this flag as appropriate for his particular application so that a chat tool need not be forced to follow a master/slave model that does not make sense. At the same time, a whiteboard application can choose to use it to control fipassin the penf among participants.

From the client side, the principal access to the system is through a web browser, which allows the control application and other Java applets in the package to be loaded on demand. Although many shared applications are written in Java, they need not be. Client applications have also been written in C, C++, and even Lisp (for a shared Emacs editor). The application program interface (API) for the TANGO system is public, and users are encouraged to port existing tools to TANGO or develop new ones as their needs require. TANGO was originally developed under a contract from the U.S. Air Force Rome Laboratory for use in a Command and Control/Emergency Management scenario, which included shared visualization of terrain data extracted from a GIS system, and other groups are using TANGO as the collaborative framework to share a variety of other applications.

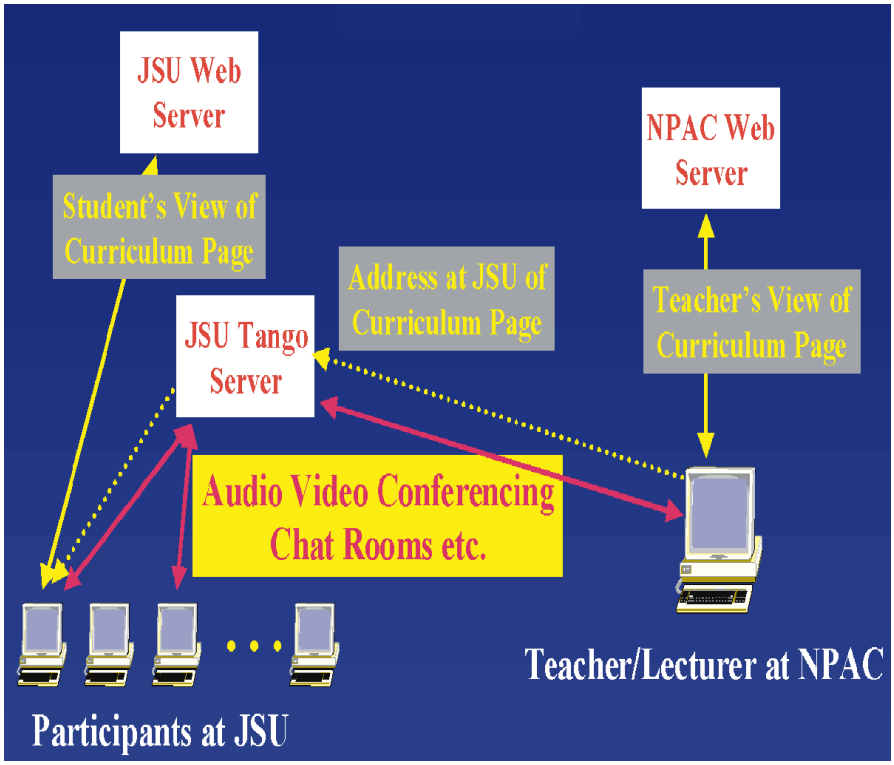


FIGURE 1. THE ARCHITECTURE OF THE TANGO-BASED DISTANCE EDUCATION PROJECT WITH JACKSON STATE UNIVERSITY. The lecturer™s shared browser uses TANGO to convey URLs for course materials to student-shared browsers, which then retrieve the page from a standard web server.

As far as TANGO is concerned, distance learning is just a special case of electronic collaboration, in which a shared web browser or similar TANGO application is used to show course materials served by a standard web server. TANGO conveys the URLs (the shared browser™s fieven sfl to the student browsers each time the instructor clicks on a hyperlink, and the student browsers respond by loading the page from the web server (Figure 1). The Syracuse faculty has used this approach to deliver three fully accredited, semester-long courses in computational science to students at PET partner Jackson State University in Mississippi, with several CEWES MSRC staff members auditing this semester™s graduate-level course at the CEWES MSRC Training and Educational Facility. The approach is also being introduced into the PET program through a series of prototype distance trainings in collaboration with the Ohio Supercomputer Center, another PET partner.

These PET initiatives are helping to transfer collaborative technologies into the DoD Research and Development community to reduce the fiimpor ance of placef in access to training, education, and general collaboration.

SPLICE: Scalable Programming Library for Coupling Executables

Brian Jean

Recent trends in large-scale computational mechanics indicate the need for a software module that will facilitate communication of data between multiple codes running simultaneously.

Sometimes, the set of interacting software programs (referred to as a ficod setfl consists of dissimilar codes that were never originally intended to function together. In addition, the codes may be running on multiple processors on different computers. This need for coupling multiple codes can occur in a number of different application areas, including fluid mechanics, environmental quality modeling, structural mechanics, and climate/weather/ocean modeling.

Researchers at the CEWES MSRC are currently building SPLICE, the Scalable Programming Library for Coupling Executables. SPLICE will have a standard communication interface that will simplify the coupling of application codes by providing the researcher with a set of intuitive, high-level library calls for establishing, maintaining, and executing external communication (Figure 1).

SPLICE will abstract each code set member from the internal structure and logic of all other members. Therefore, once a code set has been adapted to use SPLICE, only minimal changes will be necessary for each code to function in other code sets. SPLICE will communicate data via simple calls to the library, after source and target members indicate readiness to send and receive data. If any member of a code set is running in parallel, SPLICE will store the distribution of data exchange items for each member and update the communication links if a member repartitions its data.

The net effect of SPLICE will be to shield the researcher from many of the intricacies associated with external communication using current message-passing libraries.

Primary goals of the SPLICE project are to enable communication between distinct codes running in parallel, possibly on different machines, and to produce a fiuse

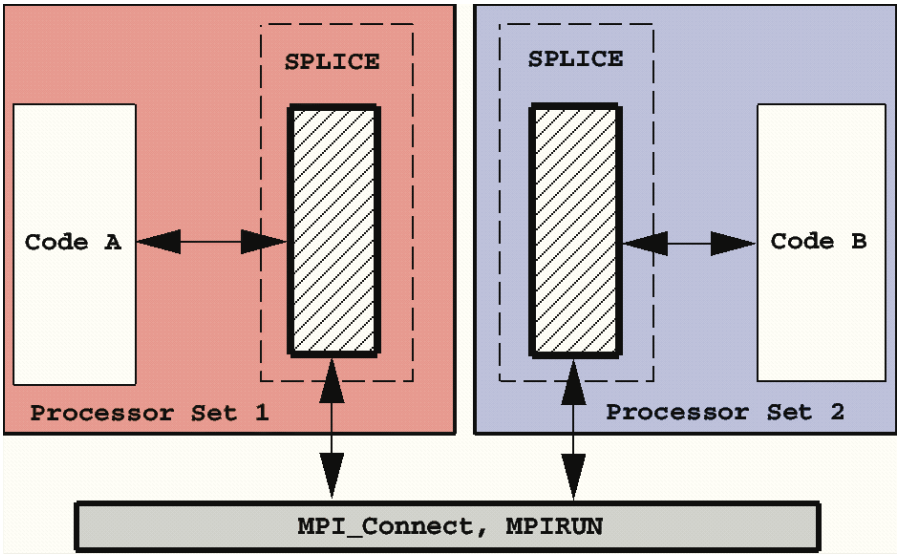


FIGURE 1. SPLICE CODE.

Programming Tools

levelf interface that will hide communication details among codes allowing a user to establish a communication link with relative ease and minimizing the modifications required for each code.

SPLICE will utilize Message Passing Interface (MPI) calls for communication. Software such as NASA MPIRUN or MPI Connect (currently under development), will be used to establish a communication protocol between codes or high performance computing (HPC) platforms. SPLICE will be object-oriented and written in C++. However, Application Programming Interfaces will be provided for both C and FORTRAN.

Initial testing of the software began during the summer 1998, with a beta release planned for November 1998.

Turbulence Modeling

Scalable Parallel Simulation of Turbulent, Noncircular Jets

S. W. Bova, Ph.D.
Fernando Grinstein, Ph.D.
Alan Stagg, Ph.D.

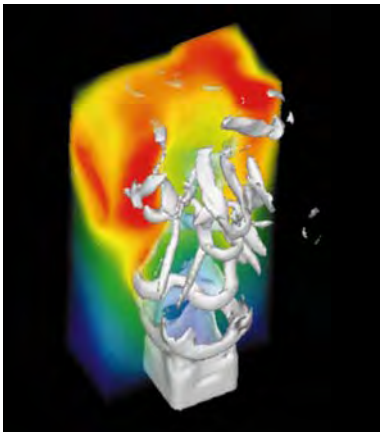


FIGURE 1. VOLUME VISUALIZATION OF THE TEMPERATURE AND VORTICITY MAGNITUDE. Instantaneous visualization of the temperature distributions (color) superimposed on isosurfaces of the vorticity magnitude (gray) for a reactive propane/air jet emerging into air background. Flow direction is from bottom to top.

The U.S. Navy is interested in improving the efficiency of jet and rocket engines used in missiles and other aircraft. Traditionally, jet exhausts have been designed using circular nozzles. Today, rectangular jets are of special interest because they have certain characteristics that may be exploited to improve the combustion process. For example, passive combustion control strategies are based on geometrical modification of the jet nozzle to manipulate the natural development of large-scale vortices and their breakdown into turbulence to enhance entrainment and mixing. If the turbulent mixing in the flow can be increased, then the reactants can be mixed more thoroughly, thereby giving more complete combustion. Compared to circular jets, rectangular jets offer passively improved mixing at both ends: enhanced large-scale entrainment due to axis-switching and enhanced small-scale mixing due to faster transition to turbulence. The code described below, NSTURB3D, simulates these processes and allows researchers to focus on understanding the dynamics and topology of the following phenomena: jet entrainment, mixing, and combustion, as well as their dependence on the aspect ratio of the jet cross section. These simulations elucidate the operative fluid dynamic mechanisms involved in the transition to turbulence and provide an improved basis for conceptual and analytical modeling of turbulent jet combustion (Figures 1 and 2).

NSTURB3D is a FORTRAN program that simulates the turbulent mixing of a compressible, three-dimensional, space/time developing rectangular jet with its surroundings. The numerical model is based on the solution of the time-dependent flow equations. In order to effectively emulate the practical flow regimes, the required simulations consume

large amounts of computer memory and CPU time. Current simulations being run on the CRAY C90 involve thousands of time-steps with 18 million unknowns per time-step. One way to accelerate large-scale simulations such as these is to use scalable, high-performance platforms, such as those available at the CEWES MSRC, which have hundreds of application processors. Furthermore, these machines have at least an order of magnitude more available memory than the C90. In order to exploit these features, a parallel, distributed-memory implementation of the existing serial code is necessary and is therefore being developed at CEWES. With such an implementation, even larger simulations can be performed in a timely fashion.

NSTURB3D uses an explicit second-order, predictor-corrector time integration scheme and flux-corrected transport (FCT) algorithm to approximate the compressible Navier-Stokes equations on a three-dimensional Cartesian grid. Orientation in the jet is expressed with respect to the streamwise and cross-stream directions. The FCT algorithm sweeps through the grid by performing a two-dimensional solve on the cross planes and a one-dimensional solve in the streamwise direction.

The parallel algorithm proceeds as follows. First, the grid and its associated data structures are statically partitioned. Since there is more computational work associated with the cross plane solves than with the one-dimensional streamwise solves, a two-dimensional decomposition of the cross plane is performed. The decomposition is further constrained to ensure load balance by specifying an approximately equal number of grid cells per processor. The resulting subdomains overlap by a layer of fighos cellsf which must be exchanged throughout the solution process by explicit communication. This communication is performed with the MPI library for portability. The use of Fortran 90 modules allows the details of the message-passing implementation to be hidden from the user and promotes software reuse and ease of maintenance. The parallel implementation is complete and is currently being verified on three CEWES MSRC scalable platforms: the IBM SP, the CRAY T3E, and the SGI/CRAY Origin 2000. After verification, simulations which require the solution of many thousands of systems of equations that have 30 to 40 million unknowns each will be performed on 50 to 100 processors.

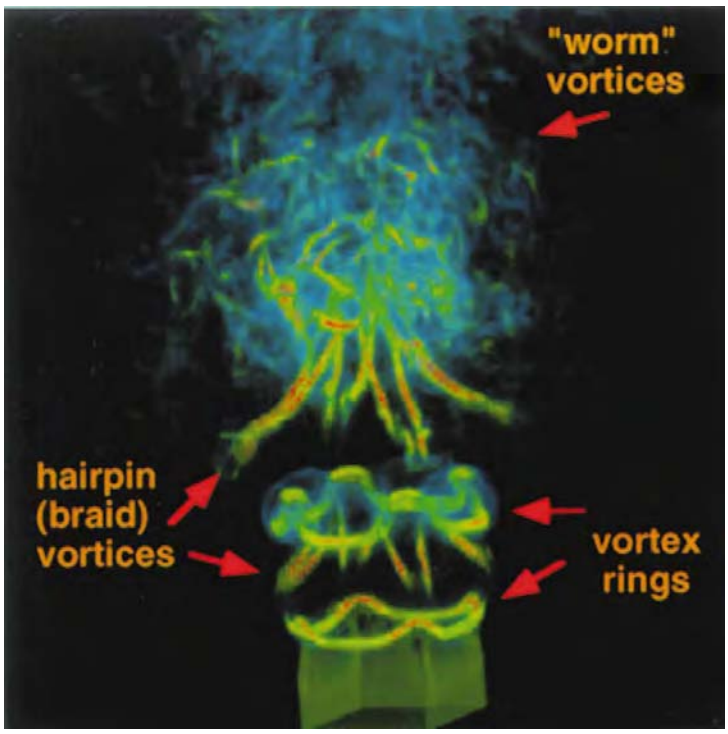


FIGURE 2. ISO-SURFACES OF THE VORTICITY MAGNITUDE. Red indicates regions of high vorticity. The square jet (at the bottom of the figure) is initially laminar and the transition to turbulence takes place along the jet axis towards the top of the figure.

Panoram System Augments High-Yield Visualization Capabilities

John E. West

As the computational resources available to users of the CEWES MSRC continue to expand, so do the complexity and size of problems solved. While very large data sets have become more commonplace in recent years, techniques for managing these data sets are only just now beginning to evolve. One problem has been that visualizations of these data sets had to be displayed on workstation monitors. Even large monitors with 1024 × 1280 pixels may have a relatively low resolution when compared with the amount of data to be displayed. This results in the need for subsampling or complex multi-resolution projection methods; however, subsampling may result in a substantial loss of information, while multi-resolution methods can be slow or difficult to implement. With the addition of the Panoram Technologies GVR-120 Reality Centre to the scientific visualization resources available at the Information Technology Laboratory (ITL), it is now possible to visualize data sets at a higher resolution than previously possible.

PANORAM TECHNOLOGIES GVR-120 REALITY CENTRE. The GVR-120 system is an arrayed video projector system.

The GVR-120 system is an arrayed video projector system. The system is driven by a Silicon Graphics Inc. (SGI) Power Onyx with Infinite Reality graphics, four R10000 processors, 3 Gbytes of RAM and approximately 30 Gbytes of disk space. The Power Onyx is connected via ATM, FDDI, and Ethernet to other MSRC and ITL resources, ensuring that data can be transferred efficiently from wherever it was

generated. Output from the Power Onyx is directed through custom hardware and three Electrohome projectors onto a 271.5-in. by 68-in. curved display. This configuration, called the MegaDesktop™, provides a 3200 × 1024 pixel viewable area to the user. In this mode the system behaves just like an extremely large monitor, providing the ability to size and move windows anywhere on the screen. Software that runs on the SGI is inherently compatible with the system and can be run without any modifications, enabling use of both custom and commercial scientific visualization software tools.

The Panoram display system supports a variety of other display modes and input devices when not in the MegaDesktop™ configuration. In addition to the SGI Power Onyx, the system is configured to accept input from various video sources, personal computers, and document display systems. Furthermore, the sources can be combined in a variety of configurations using the library of display parameters provided with the system. Thus, up to three different sources can be displayed on the screen at once, or any two may be displayed in a variety of arrangements at one time. The system can also be used to create high-resolution virtual environments. The curved display fills approximately 120 degrees of the users™ field of view and is equipped to produce stereo output when used with StereoGraphics CrystalEyes™ glasses. The conference room in which the system is located has a high-fidelity audio system for a complete sense of immersion.

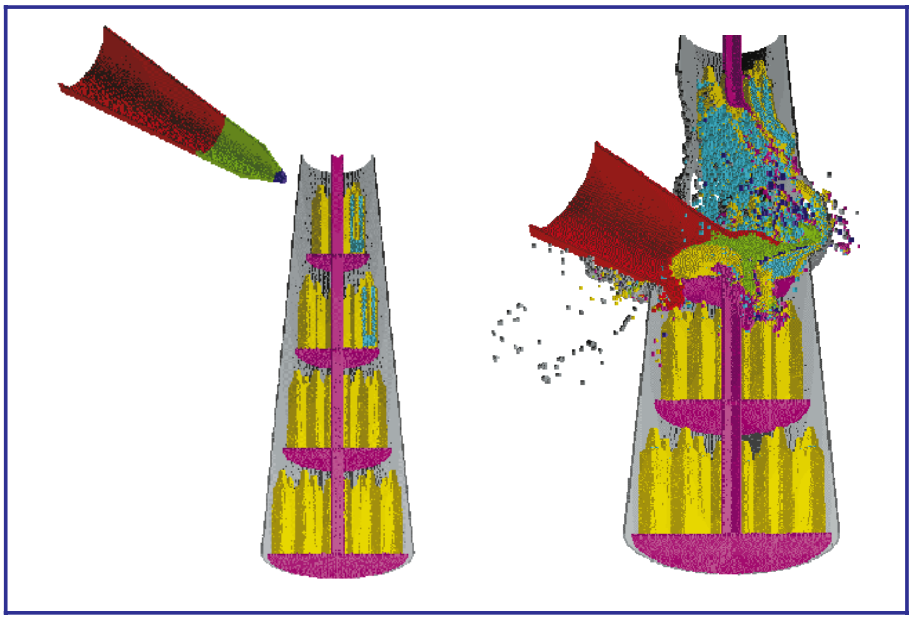
Due to its resolution and size, the display is well suited to conveying information to a large audience, and the flexibility of the system allows application designers to create visualizations for a high-yield visualization experience. Several applications recently created by the CEWES MSRC for this environment use two of the three projectors (or two-thirds of the screen) to visualize data while a companion video is displayed on the remaining projector. This configuration permits the application scientist or visualization specialist to communicate the nature of the problem being addressed in the video portion of the display area and move directly to visualization of the data without having to shift the viewers™ focus. Furthermore, the side-by-side display permits direct comparison of the results of experiments and simulations while maintaining high resolution.

The large, curved display also makes the Panoram system well suited to CADD/GIS applications, including architectural walk-throughs and assessment of engineering design alternatives at the conceptual stage. The Scientific Visualization Center staff has a long history of producing highly effective, professional quality animations of engineering concepts using industry standard packages such as Maya™ from Alias|Wavefront. Typically these models are produced originally as part of an animation production. However, a significant side benefit of the model building process in these packages has been the ability to interactively explore these models using custom applications created by the Scientific Visualization Center.

Developing a FORTRAN API to the Pthreads Library

Henry A. Gabb, Ph.D.
Clay P. Breshears, Ph.D..
S. W. Bova, Ph. D.

Pthreads is a POSIX standard established to control the spawning, execution, and termination of multiple tasks within a single process. Concurrent tasks are assigned to independent threads. At all times only a single process exists within a single address space, although multiple processors may be employed to execute the various threads. Even on a single processor computer, however, Pthreads programs are often more efficient due to better resource utilization. For example, one thread performs calculations while another handles input/output (I/O). Computation and I/O can usually be done simultaneously, even on a single processor.



THE FORTRAN API TO Pthreads was used to parallelize a high-energy impact particle dynamics code. Courtesy of U.S. Air Force Research Laboratory.

As useful as Pthreads is for parallel programming on shared-memory computers, a FORTRAN interface is not defined by the POSIX standard. However, the technical barriers to implementing such an application program interface (API) are not insurmountable. First, the library is small, consisting of only 61 routines which can be loosely classified into three categories: thread creation, termination, and manipulation; synchronization; and scheduling. The latter category must interact with the system and represents the lowest level of the library. As such, the bindings to these routines are difficult to test.

Another technical difficulty is that Pthreads makes extensive use of C structures. The data in these structures are only manipulated during calls to the library. The application program rarely needs to access the data directly. So rather than trying to pass FORTRAN-derived types into C structures, only the locations are communicated between the application and the interface. These memory addresses are declared private to the interface so that the application cannot inadvertently forget where the structures are located.

The programmer must also pay close attention to variable scope since the rules for scoping are different in FORTRAN and C. This is particularly important since each thread has its own stack space in addition to global memory. Another slight difference is that the FORTRAN interfaces with the Pthreads library

through subroutine calls instead of function calls. For example, a C thread obtains its identification code

```
my_id = pthread_self();
```

whereas a FORTRAN thread gets its ID as follows:

```
call fpthread_self( my_id )
```

Notice that the routine names are slightly different. This naming convention is similar to that of a parallel virtual machine (PVM).

The interface consists of two files. The first is a FORTRAN 90 module containing some necessary constants and structure definitions. These structures (which are really FORTRAN 90 derived types) are completely artificial; they contain memory addresses instead of data. The other file contains bindings to the Pthreads library along with the necessary include files. These bindings are written in the C programming language. The API is a useful tool for parallel programming. The FORTRAN API to Pthreads library was completed by engineers and scientists at the CEWES MSRC, and the interface, including documentation, should be available by November 1998. Anyone interested in obtaining the API should contact the Computational Migration Group through the CEWES MSRC Customer Assistance Center at 1-800-500-4722 or *http://wes.hpc.mil/*.

Tips to Improve Start-Up Time for Parallel Jobs on the IBM SP

R. Phillip Bording, Ph.D.

The CEWES MSRC team members offer the following suggestions for improving system performance.

Starting a Job That Uses Many Processors

In order for a job to start on each of the IBM SP compute nodes, a copy of the executable code must reside in a directory of a disk file system and be memory mappable (executable codes located in PIOFS-based file systems are not memory mappable). When a job is activated by a job manager, it does so from some disk file system directory. The file system directories on an SP compute node are either local ($SCRATCH) or remote ($HOME, $HOMEDIR, $WORKDIR). If the user starts a job with an executable code that resides in a remote directory, the remote high availability file server (HAFS) must deliver the blocks associated with that code, directly impacting the job load time. For SP jobs using just a few compute nodes, this start-up time is nominal. However, for jobs that use a large number of compute nodes (64 or more), the time required to move multiple copies of the executable code can be very long.

One solution is to exploit the compute node™ local file system, $SCRATCH. The executable code can be placed in $SCRATCH for the life of the job. The high-speed remote $WORKDIR file system (either GPFS or PIOFS based) is attached to all of the nodes that comprise the SP system and can

act as an intermediate transfer mechanism.

The methodology is this: perform a single remote file copy operation, copying the executable (i.e., fiprogra _binaryfl from $HOME or $HOMEDIR to $WORKDIR. Next, perform a parallel copy operation, copying the program_binary from $WORKDIR to $SCRATCH on each of the allocated SP compute nodes. Execute the copy of the program binary located in $SCRATCH on each node. Finally, and most importantly, clean up $SCRATCH so that successive jobs will have adequate disk space.

Experiments at the CEWES MSRC have shown that this method dramatically reduces the time to start large jobs. Additionally, the variation timing due to moving initial memory pages from $HOME is greatly reduced.

The following code illustrates how to perform this job staging to $SCRATCH via $WORKDIR:

```
Main_Script:
#!/bin/ksh
#PBS -l nodes=64,walltime=03:00:00
#PBS -j oe
#PBS -V
# prepare for parallel copy
cp  -p  $HOME/program_binary
     $WORKDIR/program_binary
# perform parallel tasks
pbspoe $HOME/Pbspoe_Script
# clean up
rm $WORKDIR/program_binary

Pbspoe_Script:
#!/bin/ksh
# distribute binary to individual compute nodes
cp  -p  $WORKDIR/program_binary
     $SCRATCH/program_binary
# execute binary
$SCRATCH/program_binary
# clean up
rm $SCRATCH/program_binary
```

The Main_Script is submitted to the PBS server via the fqsubf command. The Pbspoe_Script is called by the Main_script and represents those tasks to be carried out, in parallel, on the allocated compute nodes.

Programming the Cache

The SP nodes have local cache memory space that is not user programmable. This means that the user or compiler cannot issue computer instructions that directly control the cache. However, the user can benefit by understanding how the cache works.

The primary idea is to allow data movement which keeps the maximum amount of data in the cache for reuse. By keeping as much data as possible in the cache, the processor delays that are associated with memory access are greatly reduced, and program performance can be improved.

The SP has four workload and store instructions that can be used by the code generator if the cache is properly described to the FORTRAN compiler.

In compiling MPXLF FORTRAN, the following cache commands should be used with the compiler:

```
mpxlf program.f -o program -O3 -qstrict
-qarch=pwr2
     -qtune=pwr2 -qhot -qcache=as
-soc=4:cost=38:
     level=1:line=256:size=128:type=d
```

Note that all of the data specified are essential to describe the cache. Also, these data apply to the POWER2 SuperChip, which is on all the SP nodes at the CEWES MSRC.

Recent Performance Study on CEWES IBM SP Demonstrates Speed of New Chip

R. Phillip Bording, Ph.D.

A scalability and performance study of ENSAERO MPI, a NASA Ames Research Center software code used for complicated computer-generated visualizations, was demonstrated to run almost 2.5 times faster on the newer 135 MHz POWER2 processors in the IBM SP at the CEWES MSRC than on the previous 66.7 MHz processors.

fiThi speed up is due to the use of the new and expanded 135 MHz POWER2 chips used at the CEWES MSRC relative to the 66.7 MHz chips,f said Mehrdad Farhangnia, MCAT Inc., NASA Ames Research Center.

fiThi is one illustration of how DoD high performance computing centers, like the CEWES MSRC, are committed to providing the latest in computing technology to the DoD user community,f said Dr. N. Radhakrishnan, CEWES MSRC Site Manager and Director of the CEWES Information Technology Laboratory.

fiRunn ng at approximately 40 million floating point operations per second (MFLOPS) per processor, this corresponds to 9 GFLOPS performance, the highest achieved to date by ENSAERO MPI and nearly 2.5 times faster than the maximum achievable performance on the 66.7 MHz processors by this code,f Farhangnia said.

ENSAERO MPI is a parallelized, high-fidelity, multi-block, multidisciplinary code with fluids, structures, and controls capabilities developed at NASA Ames Research Center. ENSAERO MPI is capable of computing aeroelastic responses by simultaneously integrating the Navier Stokes equations and the finite element structural equations using aeroelastical adaptive, dynamic grids (Figure 1.).

The major systems making up the core of the CEWES MSRC are an IBM SP, a Silicon Graphics Incorporated (SGI) Origin 2000, a CRAY T3E, a CRAY C90, and a 100-Terabyte mass storage archival system. The center provides classified and unclassified scientific visualization services, on-site computational assistance, and a fully staffed user services department.

FIGURE 1. THE NEW 135-MHZ PROCESSOR ENABLES FASTER TURN AROUND FOR RESEARCHERS STUDYING F-18 AEROELASTIC TAIL RESPONSES. PHOTOS COURTESY OF THE U.S. NAVY.

The code is parallelized on a coarse grain level using MPI. The different disciplines are solved independently on separate nodes, with the flow

Performance

domain partitioned further into a number of subdomains. There is also multi-parameter parallelization, where various parameter sets are run concurrently for a particular configuration.

The model being studied is a wing-body-empennage configuration, which consists of a single block H O grid with 180 × 173 × 40 points in the streamwise, spanwise, and body normal directions, respectively. The grid is split into multiple, equally sized zones cut perpendicular to the streamwise direction, with each zone assigned to a separate processor. Timing functions are utilized to exclude initialization and input/output (I/O) CPU usage; thus only the solver portion of the code is represented.

The multiple parameter set plot showed the scalability of the code relative to a 9-zone case for steady flow computations. The code was scaled up to 25 parameter sets (225 nodes) on the CEWES MSRC machine. The single parameter set showed the performance of splitting of the volume grid into 18 and 36 zones. The difference in processor speed was evident here as the CEWES MSRC machine performed 50 percent faster than the 66.7 MHz processor on the 9-zone case (Figure 2).

fiOver ll, the performance of the CEWES MSRC SP has been very encouraging in both per node performance and scalability, as well as turnaround times. Production runs are now in the works for an 18 block, F-18 wing-body-empennage case, where dynamic aeroelastic responses of the tail will be analyzed,f Farhangnia said.

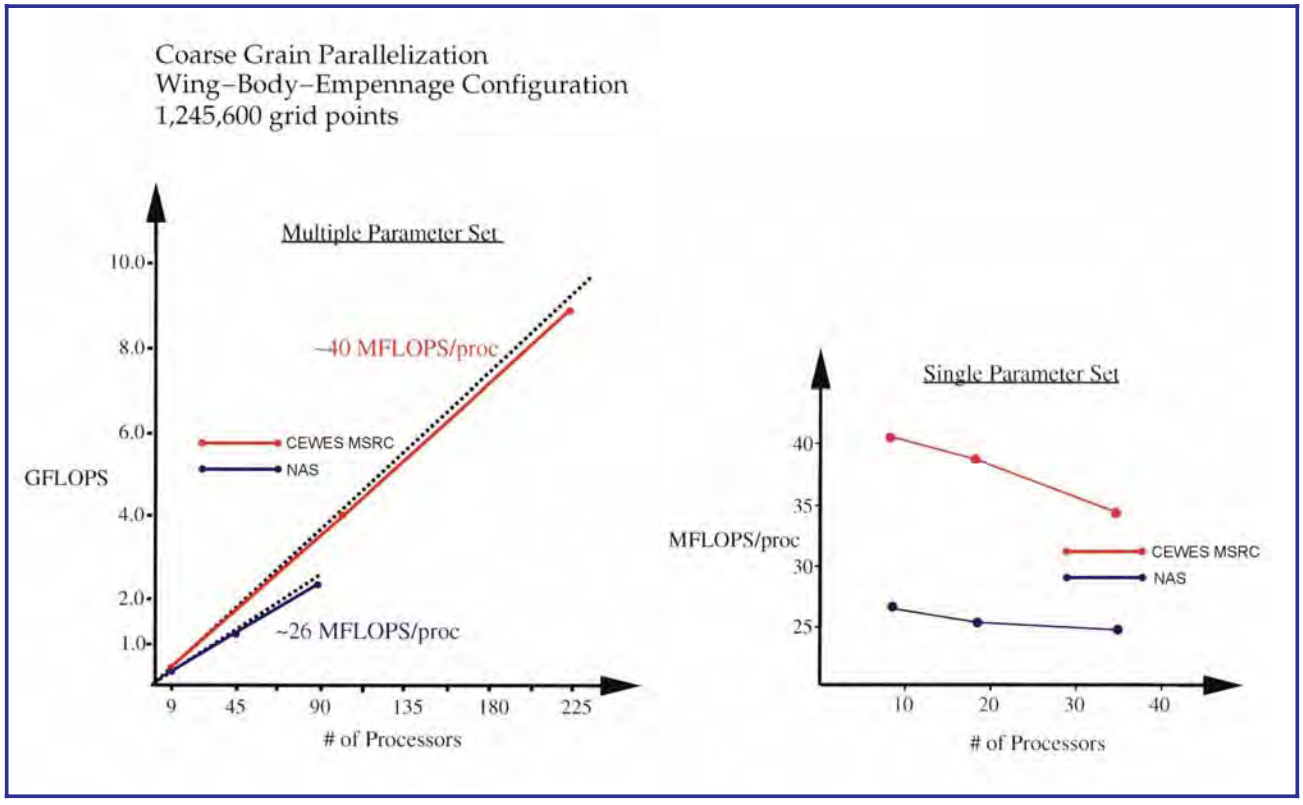


FIGURE 2. SCALABILITY AND PERFORMANCE OF ENSAERO-MPI ON THE SP.

Adapting AVS to HPC Data

Kent Eschenberg , Ph.D.

Many visualization projects can benefit from using commercial, off-the-shelf (COTS) software packages. However, some data sets, including most encountered at the CEWES MSRC, do not quite fit into the input file model supported by these packages. The CEWES MSRC Scientific Visualization Center (SVC) has developed custom input modules for the AVS/Express Visualizer package that convert the HPC data to the model used by AVS.

The Data Model

Within the AVS visualization, system data are stored as a field, a self-describing data structure and dimensions, units, labels, coordinates, and data values. A field can be structured (with various levels of coordinate detail) or unstructured. An unstructured field may contain any number of arrays of cells, called cellsets, where each cellset is a specific geometry type: point, line, triangle, quadrilateral, prism, pyramid, tetrahedron, and hexagon. One or more data values (for example, salinity) can be fiattached to the vertices, the cell center, or both.

The custom input modules described below have a simple goal: read a data file and produce a field. Each module is designed to read a specific type of file and produce a specific type of field. The actual visualization modules (such as those that create isosurfaces and cut planes) are designed to work with most types of fields; therefore, a user can work with the same visualization options with many different types of fields.

Working with Time-Varying Data

Data sets resulting from simulation on high performance computers vary over time. Early versions of AVS did not support time-varying data. The version currently in use at CEWES MSRC, Visualizer 3.4, contains several problems with time-varying fields. More recent versions of the software may provide excellent support for time-varying data.

Thus, a locally developed approach has been used for the management of time-varying fields. Each input module contains three subroutines: the fireader, which reads the input files and saves all of the data in memory; the fislicer, which extracts the data at one particular time and creates a field; and the fireleaser, which frees the memory used to store the data when the module is deleted.

The reader is often enhanced to provide some other services. For example, some readers include fifilters that can be set by the user to eliminate certain categories of data before they are even stored in memory. This can be critical to working with some of the very large HPC data sets. Many visualization systems (including AVS) implement such a filter by first reading everything into memory and then making a copy of the parts of the field that were of specific interest.

Another example of a special service provided by the reader is interpolation to provide finer time-steps or to smooth out irregularly spaced time-steps. While many visualization packages can perform this sort of function as they step through time, the approach used in the SVC custom modules is to perform the interpolation once, during the read phase, instead of every time the interpolated time is accessed.

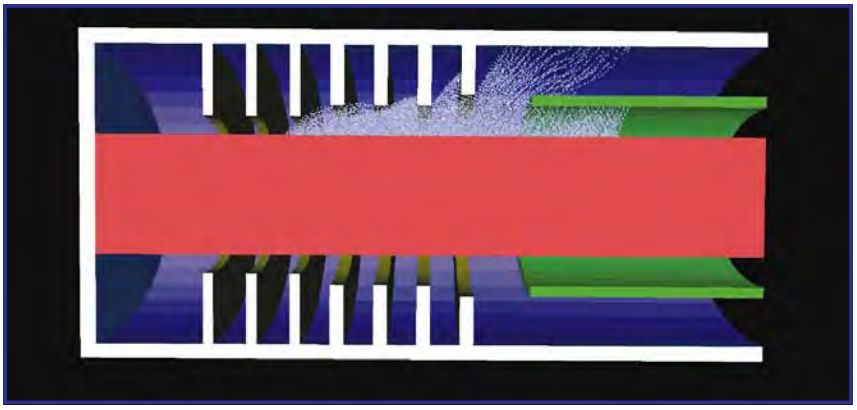


FIGURE 1. Particle flow for the RFW visualization at a time-step soon after the system has been started.

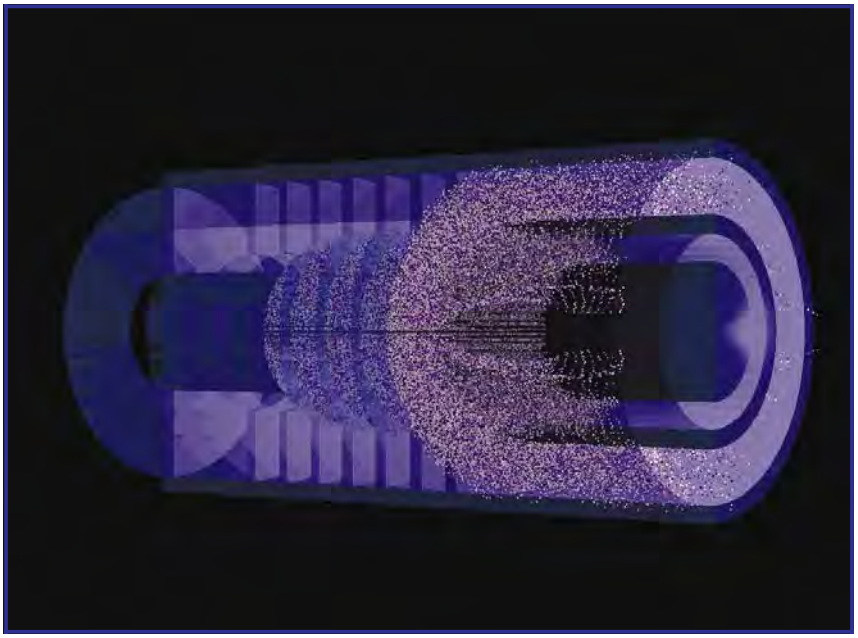


FIGURE 2. The full three-dimensional particle set from the RFW simulation with transparent structures to improve the view of interior areas.

Example Projects

The Radio Frequency Weapon (RFW) project is using HPC simulations to develop high power microwave emitters. The simulation tracks the location of electrons as they are emitted and oscillated inside the device. The raw data files can occupy as much as 2 gigabytes of storage on the HPC disk. A data reduction scheme was developed to reduce the size of this data set before it could be visualized. First, a custom preprocessor on the HPC filters and condenses the data into a special file format; second, a custom AVS input module reads this file. The field produced by this module consists of unstructured points that can be displayed as pixels or spheres (Figures 1 and 2).

After the RFW input module was completed, two other CEWES MSRC projects emerged that were very similar. In one case, the salinity of the water in a bay at about 500 time-steps needed to be analyzed. In the other case, the probability of structural failure as a function of three variables and time needed to be viewed. In both cases, a new custom input module was developed by making modest changes to the RFW input module.

Conclusion

Using COTS visualization packages can greatly reduce the time needed to complete a project and can help improve the quality of the results by allowing the developers to work with the similar visual options from project to project. With the right approach, writing custom input modules can be an efficient and elegant solution with a large amount of module reuse.

Preprints

Preprints are available from the CEWES MSRC Customer Assistance Center or on the CEWES MSRC web site at *http://wes.hpc.mil/pet/CEWES/CEWES_reports.html*. For further information, contact the Customer Assistance Center by telephone at 800-500-4722 or by e-mail at *info-hpc@wes.hpc.mil*.

Forces Modeling and Simulation

98-20 fiJWOR - Java Web Objects Request Broker for Commodity Software Based Visual Dataflow Metacomputing Programming Environmentf

98-21 fiBu lding Web/Commodity Based Visual Authoring Environments for Distributed Object/Component Applications - A Case Study Using NPAC WebFlow Systemf

98-22 fiExplorin JSDA, CORBA and HLA Based MuTech™ for Scalable Televirtual (TVR) Environmentsf

98-23 fiIntegratin Web, Desktop, Enterprise and Military Simulation Technologies to Enable World-Wide Scalable Televirtual (TVR) Environmentsf

98-34 fiWeb LA - An Interactive Programming and Training Environment for High Performance Modeling and Simulationf

98-43 fiMicrosof DirectPlay meets DMSO RTI for Virtual Prototyping in HPC T&E Environmentsf

98-44 fiObjec WEB (Java/CORBA) Based RTI to Support Metacomputing M&Sf

98-46 fiEva uating New Transparent Persistence Commodity Models: JDBC, CORBA PSS, OLEDB and W3C WOM for HPC T&E Databasesf

Computational Fluid Dynamics

97-02 fiSom Performance Issues Associated with CEWES MSRC Scalable Architecturesf

98-03 fi Fortran90 Module for Message-Passing Applications with Unstructured Communication Patternsf

98-07 fiO the Parallelization of CH3Df

98-09 fi Computational Design Method for High-Velocity Channelsf

98-49 fiStatu Report on Parallelization of MAGIf

Enviromental Quality Modeling

98-10 fiPCE-QUAL-ICM A Parallel Water Quality Model Based on CE-QUAL-ICMf

98-11 fiProgres Report: Parallelization of ADCIRC3Df

98-19 fi Projection Method for Constructing a Mass Conservative Velocity Fieldf

Computational Structural Mechanics

98-28 fiGri Generation Capabilities Enhancement at the CEWES MSRCf

98-36 fi Representaive Survey of Blast Loading Models and Damage Assessment Methods for Buildings Subject to Explosive Blastsf

Climate/Weather/Ocean Modeling

98-14 fiWA Performance Improvement and NLOM Optimizationf

98-15 fiCo pling of Circulation, Wave and Sediment Modelsf

Scalable Parallel Programming Tools

98-02 fiRevie of Performance Analysis Tools for MPI Parallel Programsf

98-05 fiUsin the MPE Graphics Library with Fortran90f

98-08 fiSoftwar Repository Interoperation and Access Controlf

98-25 fiTh CacheBench Reportf

98-26 fiTh MPBench Reportf

98-27 fiTh BLASBench Reportf

98-31 fiMPIC on the Cray T3Ef

98-32 fiToward a Fortran 90 Interface to the POSIX Threads Libraryf

98-33 fiScaLAPAC Evaluation and Performance at the DoD MSRCsf

98-38 fi Fortran 90 Application Programming Interface to the POSIX Treads Libraryf

98-39 fiPerformanc Evaluation of HPF Kernels on the IBM SP and CRAY T3Ef

98-42 fiMP Interconnection and Controlf

Scientific Visualization

98-12 fiComp ring Middleware Support for Collaborative Visualizationf

98-16 fiVis alization of Damaged Structuresf

98-17 fiStructur Significant Representation of Structured Datasetsf

98-18 fi Numerical Imaging Approach to Comparative Visualizationf

98-24 fiVisGe Cross-Platform Visualizationf

98-30 fiIdent fying Boundary Anomalies to Facilitate Correct Parallel Image Compositionf

98-48 fiVirtua Prototype Radio Frequency Weapon Project Visualization: A Case Studyf

Collaboration and Communication

98-13 finetWor Place Project Reportf

98-41 fiCE ES Database Projects Developed by NPACf

98-45 fiRea Time Training and Integration of Simulation and Planning using the TangoInteractive Collaborative Systemf

98-47 fiTA GO Interactive for Remote Consulting and Group Software Developmentf

Training

98-01 fi199 CEWES MSRC PET Training Activitiesf

98-04 fiSurve of the Construction and Utility of PET Virtual Workshopsf

98-29 fiSynchronou Learning at a Distance: Experiences with TANGOf

Miscellaneous

97-01 fi Taxonomy of Major CTA Software at CEWES MSRCf

98-06 fiContrac Year Two Programming Environment and Training (PET) Additional Focused Efforts for CEWES Major Shared Resource Center (MSRC)f

98-35 fiUpdat to Taxonomy of Major Computation Technology Area (CTA) Software at CEWES Major Shared Resource Center (MSRC)f

98-40 fiContrac Year Three Programming Environment and Training (PET) Core Support and Focused Efforts for CEWES Major Shared Resource Center (MSRC)f

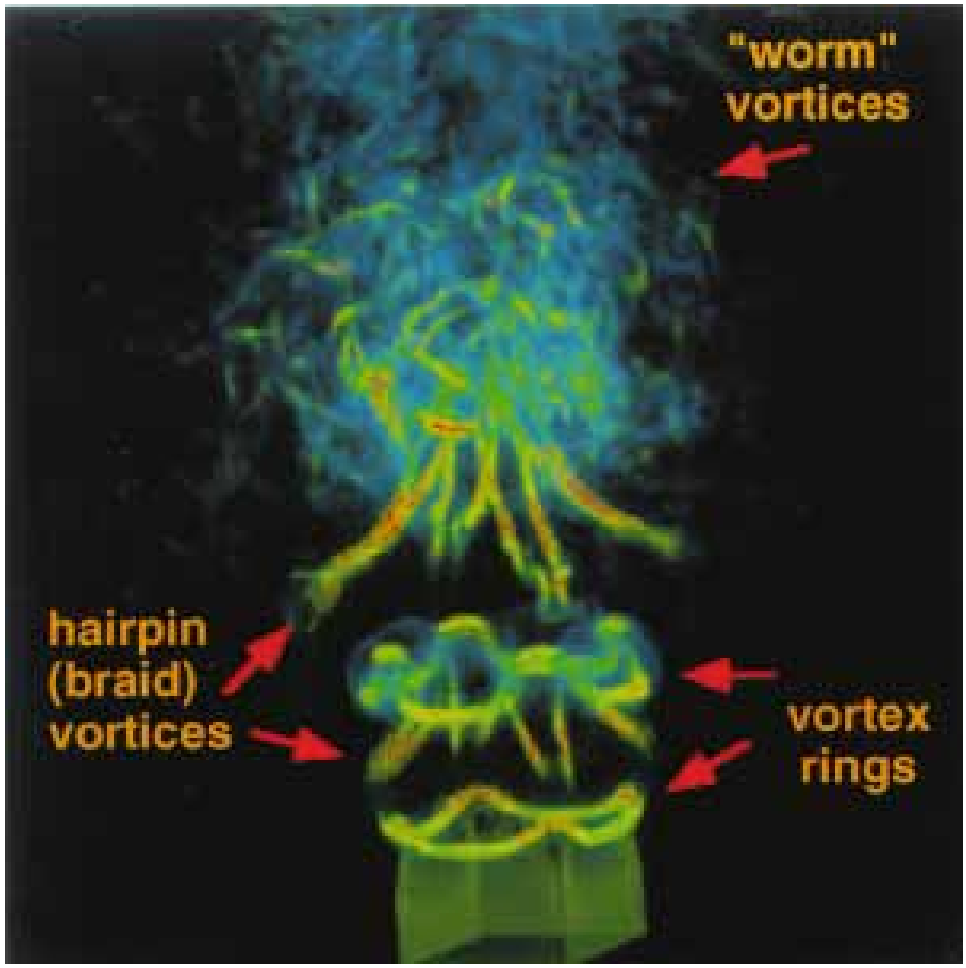
For further information, contact:

CEWES MSRC
Customer Assistance Center

Web site
http://wes.hpc.mil

E-mail at *info-hpc@wes.hpc.mil*

Telephone at 800-500-4722



VORTICITY DYNAMICS IN A SQUARE AIR JET EMERGING INTO AIR BACKGROUND

Improving the mixing of a jet (or plume) with its surroundings is of considerable interest in practical applications demanding enhanced performance of combustors in missile and other Navy aircraft propulsion systems. Geometric modifications to a jet nozzle can efficiently provide such improvements by directly affecting the formation of large-scale vortices and their breakdown into turbulence. The subsonic square jet in the figure evolves from laminar initial conditions (bottom of the figure). A ray-tracing technique is used to visualize the vorticity magnitude, ranging from semi-transparent blue to opaque red. The square jet development is characterized by the dynamics of strongly-interacting vortex rings and hairpin (braid) vortices in the near jet, and by elongated fiw rmf vortices in the turbulent region downstream. This simulation involved nearly 10 million grid points. Performing larger simulations demands distributed-memory implementation of the existing serial code to scalable, high-performance platforms.

U.S. Army Engineer Waterways Experiment Station
ATTN: CEWES-IH/CEWES MSRC
3909 Halls Ferry Road
Vicksburg, MS 39180-6199